A Configurable Packet Classification Architecture for Software-Defined Networking

A Configurable Packet Classification Architecture for Software-Defined Networking

K. Guerra Pérez, X. Yang, S. Scott-Hayward, S. Sezer
The Institute of Electronics, Communications and Information Technology (ECIT)
Queen's University of Belfast, UK

***Abstract*— Network management tools must be able to monitor and analyze traffic flowing through network systems. According to the OpenFlow protocol applied in Software-Defined Networking (SDN), packets are classified into flows that are searched in flow tables. Further actions, such as packet forwarding, modification, and redirection to a group table, are made in the flow table with respect to the search results.**

A novel hardware solution for SDN-enabled packet classification is presented in this paper. The proposed scheme is focused on a label-based search method, achieving high flexibility in memory usage. The implemented hardware architecture provides optimal lookup performance by configuring the search algorithm and by performing fast incremental update as programmed the software controller.

Keywords—Packet Classification; Software-Defined Network; configurable lookup architecture; lookup algorithms;

I. INTRODUCTION

The exponential growth of new network applications and services, such as virtual machine usage, is overloading network device resources. Software-Defined Networking (SDN) has arisen as a platform to reduce the complexity of network elements.

The SDN platform manages traffic loads in a flexible and more efficient manner by splitting software and hardware resource controls.

One of the great benefits of the SDN architecture is the ability to direct traffic through the network on a flow basis. This enables network service function chaining, for example, where flows are directed through a series of network services depending on the traffic or application type [1].

Packet classification is the main part of flow identification by which the action for each incoming flow at a network device is determined. This action is determined by the Highest Priority Matching Rule (HPMR) from a given filter. It is only necessary that the first packet header of a flow matches the matching rule.

In general, five tuples from packet headers are used for classification: protocol, destination and source ports and source and destination addresses from Layer 3-4 of the Open System Interconnection (OSI) model.

For the next-generation network, packet classification must support high network throughput, e.g. 40-100 Gbps, a wider range of packet fields and consequently a large rule set. It is well known that certain parameters such as scalability, flexibility, capacity, incremental update ability, memory usage and speed, are used to measure the efficiency of lookup systems.

Our hardware implementation based on a configurable search algorithm is well-suited to the programmable platform of SDN, providing greater flexibility than previous methods.

The rest of the paper is organized as follows. In section II, the background to the work is introduced and the related work is discussed. The proposed architecture is presented in section III and the design methodology is described in section IV. In section V, performance evaluation results are presented and discussed. Finally, in section VI, we conclude the paper.

II. RELATED WORK

The process of organizing packets into flows uses multiple fields of the packet header. Each of these fields is defined in diverse syntaxes, such as ranges or prefixes. As a result, each field requires a different matching method, for example Exact Matching (EM), Range Matching (RM) or Longest Prefix Matching (LPM). This presents a challenge to existing packet classification algorithms. In addition, with increasing granularity of the flow definition, an increasing number of flow match entries are held in the flow tables of network devices.

Ternary Content Addressable Memory (TCAM) is a popular method for classification due to its high lookup speed. However, this technique is tending to be replaced owing to disadvantages of high power consumption, storage limitation and the difficulty of rule ternary conversion.

Several algorithms have been proposed for packet classification in which each input packet header must be classified by comparison with a given rule set and processed according to a defined action. These packet classification algorithms can be classified into two categories: multi-field lookup algorithms and single-field lookup algorithms.

Recent research proposed improvements to packet classification efficiency based on the most popular methods, such as HyperCuts [2] or Recursive Flow Classification (RFC) [3]. RFC is of interest due to its high speed performance [4]. Distributed Cross-producting Field Labels (DCFL) [5] is a decomposition method in which individual-field lookups are performed in parallel. The individual results are combined to produce the final result using a label method. Although the lookup performance is high, the memory utilization is inefficient. A DCFL Extended (DCFLE) technique is presented in [6]. This methodology used extended TCAMs (ETCAM) and TCAMs for comparison in a reconfigurable hardware architecture.

Recent research on SDN is focused on conventional packet classification methods adapted for next generation networks.

The EffiCuts algorithm presented in [7] is based on HyperCuts. EffiCuts reduces memory space by reducing the number of overlapped rules but with increased memory access time. The same algorithm was proposed in [8] based on OpenFlow [1]. The authors group rules according to their sizes for efficient storage. The work presented in [9] proposed a HyperCuts based algorithm. These two algorithms require a large number of stored nodes and need a computing stage for multi-dimensional cutting.

A decomposition-field approach is evaluated in [10] on a multi-core processor based on OpenFlow. This research performs parallel field search using a balanced range tree or hash function. The search result from each field is stored in a Bit Vector and the final result is obtained by computing each bit vector. Although this method is proposed to handle a large number of fields, it is not suitable for high-speed lookup in current network systems. Furthermore, the authors do not provide evaluation results for the update operation, which is a key process in packet classification.

III. PROPOSED ARCHITECTURE

The SDN architecture consists of three layers/planes: Application, Control and Infrastructure/Data plane [1]. This architecture is illustrated in Fig. 1. Note the distinction between a network application, which provides instructions to the control plane (e.g. load-balancing), and a user application, which determines the packet/flow type across the network (e.g. video). For the purpose of this work, we consider the user application with respect to the traffic to be classified and the network application with respect to the rules, which determine how to handle the user traffic. The rules generated at the controller are pushed to the network devices by means of an open protocol such as OpenFlow.

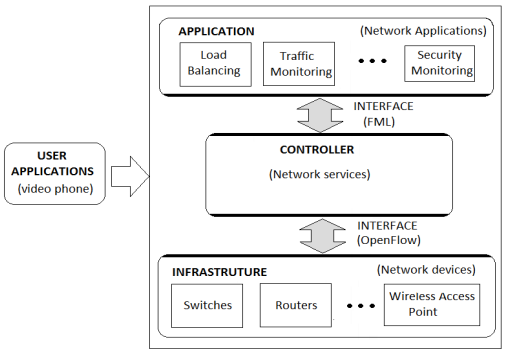

Fig. 1. Software-Defined Network Structure

Two different approaches, multi-field and single-field lookup, were analyzed using the most popular packet classification algorithms in our previous work [17].

Table I summarizes the results from the comparison analysis. The most popular algorithms based on different approaches are evaluated in order to compare them against the measures of memory usage and average number of lookup memory access. Lower memory usage is preferable and a lower number of memory accesses implies a higher lookup speed. HyperCuts and RFC are considered as the most popular multi-field lookup algorithms in two different approaches: decision-based tree and decomposition. DCFL focuses on another approach based on the combination of one-dimensional lookup algorithms. Finally, two options are considered as the best results from the comparison between different levels of trie based algorithms. These options are based on a combination of one-dimensional algorithms according to the syntax required for each field. Thus, Option 1 is represented by a 5-level Multi-bit trie, a 4-level Segment trie and a register-based lookup table for IP address fields, port fields and protocol field respectively. Option 2 is formed by a 4-level Multi-bit trie, a 5-level Segment trie and a register-based lookup table.

TABLE I. PERFORMANCE EVALUATION OF ALGORITHM BASED ON DIFFERENT LOOKUP APPROACHES

Algorithm	Lookup performance (Avg. Memory access number)	Memory Space (Mb)
HyperCuts [2]	60.05	5.96
RFC [3]	48	31.48
DCFL [5]	23.1	22.54
Option 1	49.3	5.57
Option 2	31.33	6.36

While multi-field lookup algorithms are used in the majority of packet classification systems, single-field lookup algorithms in parallel produce more successful results in addressing issues such as lookup or memory space. For example, RFC requires 48 memory accesses for lookup process as the best case of multi-field algorithm, whereas DCFL requires 23.1 memory accesses on average, providing higher lookup speed. Furthermore, the analysis in this research demonstrates the fact that combinations of single-field based algorithms provide an optimal solution for packet classification.

Our proposed system takes advantage of the flexibility and programmability introduced by SDN. A hardware design based on the optimum lookup approach for a given application is proposed in this paper. It is then possible for the appropriate lookup process to be selected by the Controller. This configurable architecture is explored in this section from the programmability perspective, which is represented in Fig. 2.

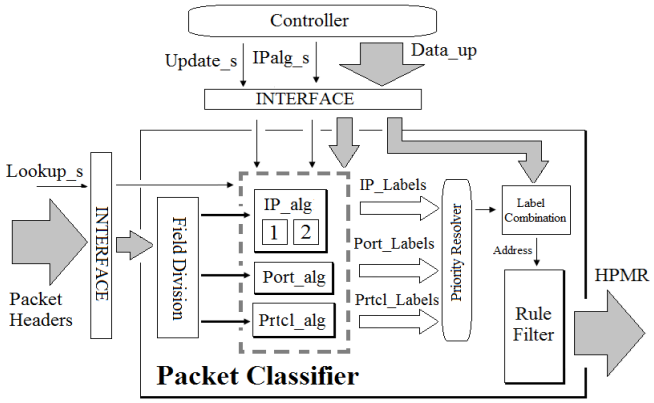

Fig. 2. Hardware Architecture of Classifier based on SDN Programmability

A. Controller Functionality

The algorithm structure requires to be updated incrementally to add or delete rules. In our proposed system, the update process is controlled by software (SDN Controller). The architecture is configured with selected algorithms for each dimension according to the specifications of the application. For example, speed is the critical parameter for a Multi-end videoconferencing application supporting real-time connection [11]. The software controller chooses the optimal algorithm combination. In our system, this decision is to select between two possible IP algorithms, which are controlled by a simple signal shown in Fig. 2 (*IPalg_s*).

Subsequently, the software controller transmits the required information to configure the relevant memory blocks of the hardware platform.

B. Lookup Process

Four pipelined phases are identified in the lookup process. The first phase is stimulated by the *Lookup_s* signal, which enables the search process in each algorithm. At the same time, the packet header is split into segments, which are sent to the corresponding algorithm selected by the software.

In the second phase, the selected algorithms perform parallel lookup. The result from each algorithm is a pointer to a list of matching labels, which are passed to the next phase.

The third phase shows the combination of the result labels in order to find the HPMR address. This combination is the product of the highest priority label stored in the first position in the list of each output algorithm.

The last phase is to access the memory and obtain the HPMR. The HPMR and, consequently, the associated action are determined by the priority label. Fig. 3 shows the lookup process phases in more detail.

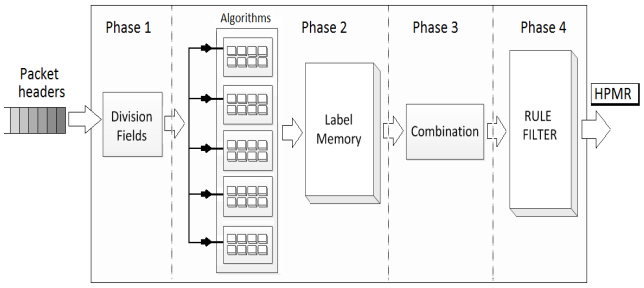

Fig. 3. Lookup Process Pipelining

C. Label Method

The label-method proposed in [5] tags each unique rule field avoiding rule duplication. Table II shows an example of the number of unique rule fields. In this case, three ACL filters [12] with different sizes are analyzed for each independent field. Avoiding rule field repetition, the storage requirement can be reduced by more than 50%. The label method is an efficient technique for algorithms with fixed structures such as Multi-bit Trie (MBT), and is not applicable to dynamic structures, which require a re-built algorithm structure.

TABLE II. NO. OF UNIQUE RULE FIELDS PER RULE SET

Packet Header Field	acl1 1K (916 rules)	acl1 5K (4415 rules)	acl1 10K (9603 rules)
Source IP Address	103	805	4784
Destination IP Address	297	640	733
Source Port	1	1	1
Destination Port	99	108	108
Protocol	3	3	3

That is, the algorithm must reorganize the trie nodes when a new node is created or deleted. This technique reduces the memory space by avoiding replicated rules in the trie.

Taking into account the fact that the single-field lookup algorithms yield better performance, the next logical step is the design of a configurable hardware architecture harnessing this optimized packet classification system for SDN.

D. Memory Blocks

Three different memory blocks can be identified in the proposed architecture: *Algorithms*, *Labels* and *Rule Filter* memory blocks. A set of memory blocks are shared among the selected algorithms.

Once the IP labels are determined, the final result is produced, incorporating the Port and Protocol labels attained in earlier stages, which are stored in storage-capacity buffers.

IV. DESIGN METHODOLOGY

A. Update Methodology

A set of binary files are created using C++ with the data needed for the hardware architecture, simulating a control plane of SDN. All cited algorithms are implemented in order to obtain hardware information, such as addresses, data nodes and label list of nodes.

The algorithm combination focuses on obtaining the optimal lookup performance required for each field. Unlike the majority of lookup algorithms, such as TCAM or EffiCuts, the conversion and adaptation of the different rule fields to a lookup method with specific syntax is not required.

The proposed work focusses on the field rule repetitions, providing high flexibility by grouping the single field rules. Thus, when one or more new rules must be inserted in the system, the Controller searches the unique labels for each field in lookup tables (*Label Tables*). The label tables also contain a counter for each label to support fast incremental update. When a label is not found in the table, three steps are performed: a new label is created, the counter is incremented by 1 and the new rule information is inserted.

However, if the label is found in the *Label Table*, only the incremental value of the counter is required. Rule deletion is performed according to the same process, deleting the corresponding labels from the table, and only when the counter is zero, the label is deleted from the hardware architecture.

The lists of labels are reorganized according to the priority rule in order to ensure the highest priority matching label (HPML) is in the first position in the list.

The final address to store each rule in the *Rule Filter* block is performed using a hash function implemented in hardware. A pseudo-code for the design of hardware information files for each algorithm is presented in Fig 4.

This information is based on available on-line filter sets [12] Access Control Lists (ACL), Firewalls (FW) and IP Chains (IPC) with a range of rule sizes, as summarized in Table III.

```
Input: binary rule file
Table.label
Table.field
for (Table.field (size)) do
    if (field == Table.field) then
        counter ++
    else
       //new label creation
       counter ← 1
       Table.label ← new label
      // chose algorithm information
       result2← lookup algorithm(field)
       result1 ← new label
    end if;
 end for;
Output: result1 and result2 are sent to the system
```

Fig. 4. Pseudo-code of Rule insertion

TABLE III. ANALYSIS OF RULE FILTERS

Filter Types	Names		
	1 K rules	*5 K rules*	*10 K rules*
ACL	916	4415	9603
FW	791	4653	9311
IPC	938	4460	9037

B. Lookup Methodology

Multi-bit Trie (MBT) and Binary Search Tree (BST) algorithms are currently used for IP lookup. The aim of this work is to build a configurable platform choosing between fast IP lookup algorithm (MBT) and efficient-memory-space algorithm (BST).

Multi-bit trie is a special tree structure where the trie depth is defined by the number of bits stored in each level and the branches are characterized by a fixed number of wildcards and exact match nodes. BST is a binary data structure where the left branches contain lower values than the right branches. The tree depth is defined by input prefixes.

It can be seen in Table II that the number of unique rule fields for Port is low. These label search methodologies are explained in section D.

Flow table lookup in SDN requires between 11 and 15 fields from packet headers. However, currently, large real-life rule sets are available with no more than five fields for analysis and performance. The authors in [8] and [9] created a scenario with 12-field rules, which has not been practically applied, while there is no evidence of the validity of the 15-field rules used in [10].

The work presented here focusses on IP field algorithm configuration as the IP address field, due to its length, potentially becomes the bottleneck in lookup performance.

C. Memory Management

In this section, the architecture is explained in more detail. In terms of memory allocation, all single-field lookup algorithms are implemented in hardware. The major challenge identified is to handle different algorithms without memory explosion. In the Algorithm memory block, a simple Look-Up Table is utilized for Protocol. The protocol value addresses the table where the label is contained.

Registers utilized for Port field lookup contain information about the port values defined in range, high value and low value of port field rule, and the corresponding label. Table IV shows an example of exact matching and range matching of port values. Each unique range is tagged with a unique label.

TABLE IV. EXAMPLE OF PORT FIELD AND LABELING

Port field rules *High value - Low value*	**Label**	**Match method**
[65355 - 0]	A	Range matching
[7812 - 7812]	B	Exact matching
[7820 - 7810]	C	Range matching

This architecture partitions the IP address field into two 16-bit segments. In other words, two MBT (and BST) algorithms are implemented, one for the high 16-bit source (or destination) IP address field and the other trie for the low 16-bit source (or destination) IP address field.

Between the two selected IP lookup algorithms, MBT supplies faster lookup. Each MBT algorithm is composed of three memory blocks corresponding to the three levels using 5-bit, 5-bit and 6-bit partitions. A pipelining MBT structure is designed improving the search throughput.

In contrast, BST is implemented in order to achieve more efficient memory usage. Therefore, a simple memory block is designated for each 16-bit segmented IP field.

The assumption is that not all tree/trie nodes will be stored in the memory so that a reduction in storage is possible. The data node contains children node pointers, a counter of labels stored and the pointer to the list of labels.

The IP lookup tree algorithms are implemented in unbalanced memory distribution. Although the lookup process accesses a large memory in some cases, the main limitation is the update process. It can be seen that the number of memory accesses and memory size increase in proportion to the number of tree nodes for rule insertion, which results in sub-optimal memory consumption. However, this methodology implies re-built structure. The perfect balanced memory cannot be achieved. Additionally, a balanced tree algorithm can be easily implemented in software and the information with the new structure can be applied in the architecture for each rule insertion.

1) Label Methodology: The label sizes are 13 bits, 7 bits and 2 bits for IP address, Port and Protocol fields respectively. The label sizes support the number of unique rule fields shown in Table II (e.g. 108 unique destination port fields can be represented by 7 bits).

It is important to maintain the labels stored in priority order in the memory. The priority is included in the given

information with rule and mask fields. This priority value defines the order in the IP label list deposited in the *Label* memory block. The priority of Port labels is given by exact matching label following by the tightest range matching label. Using Table IV in an example, for an input packet with a destination port field equal to 7812, the labels of Port lookup will be ordered as **B**, **C** and **A**. The priority label for Protocol lookup is determined by the exact matching value.

According to the order, the first label in the list of each lookup algorithm corresponds to the highest priority matching rule address. The first labels are merged in one large data segment (68 bits) in which a hash function is used to obtain the HPMR address.

2) Memory Sharing: The information provided by the software corresponds to all the data that must be stored in a specific address for an explicit memory including the three main memory blocks.

Using the label method, the *Rule Filter* memory block is treated independently of the chosen algorithm. Moreover, the *Label* memory block for one field can also be stored without any effect on the chosen algorithm combination. However, any single-field lookup algorithm must obey the following condition:

> *As the label represents a completed or segmented unique rule field, the packet header must be split into equal size segments for any chosen algorithm.*

The benefit is that, the HPML lookup methodology is maintained for any chosen algorithm combination using a simple signal (*IPalg_s)* to enable one precise input data.

As both IP lookup algorithms are implemented in the hardware architecture, there is no benefit from memory space reduction due to the fact that the hardware synthesis must consider all the different memory blocks for each algorithm. Nevertheless, this research proposes a new approach by sharing memory blocks. The main advantage is to use the same memory for two different input data depending on the selected algorithm. In this system, the MBT level-2 memory requires the same characteristics of dimension and output and input size as the simple BST memory. The rest of the memory determined for MBT can be used to collect more rules. The decision about which data is stored is determined by *IPalg_s* signal. The consequence is flexibility of memory storage and with efficient allocation data. Fig. 5 represents an example of memory sharing for one segmented IP field.

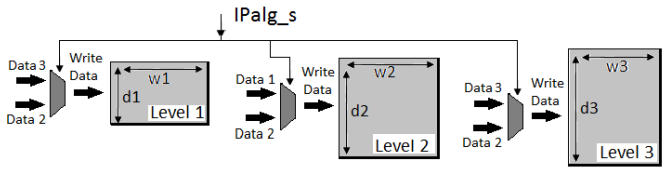

Fig. 5. Memory sharing diagram

Data 1 and Data 2 represent the node information corresponding to MBT and BST respectively and Data 3 represents the rule information. According to the *IPalg_s* signal the node information of Data 1 or Data 2 is stored in the Level 2 memory and Data 3 or Data 2 is selected to store in the rest of the memory block.

V. Performance Evaluation

A. Memory Accesses For Update

One of the benefits of this architecture based on SDN is fast update. Rule Insertion and deletion are achieved by a simple memory upload in two clock cycles per rule; one cycle to store source information and one clock cycle to store destination information due to the limitations of number of input-output pins available. As previously mentioned, an additional clock cycle is required to obtain the rule address using hash function implemented in hardware.

B. Memory Accesses For Lookup

Protocol and Port field methods realize very fast lookup. The protocol label search is executed in a single clock cycle. The Port lookup process produces the labels in two clock cycles.

The MBT structure is executed in the pipelining stage and, thus supports high throughput. MBT performs the lookup in the latency of 6 clock cycles. As expected, the BST algorithm requires a large number of clock cycles per input packet.

The algorithms provide the pointer to labels demanding one more cycle for the entire lookup process.

Finally, two more cycles are required for the final result processing in each case. This final phase is pipelined.

C. Memory Consumption

The proposed architecture was implemented on Altera's StratixV FPGA which contains a memory capacity of 54 Mbits. The memory usage of the designed architecture consumes 4% of total memory for the mentioned algorithms, lists of labels and the rules filter. The proposed circuit is able to operate 42.73 Gbps with 40 bytes per packet as minimum size, overcoming the OC-768 rate. This is not achieved in software-based systems such as [8] and [10]. Table V summarizes the synthesis results.

TABLE V. Synthesis result on Altera Stratix V device (5SGXMB6R3F43C4)

Logical Utilization	79,835 / 225,400
Total block memory bits	2,097,184/ 54,476,800
Total registers	129,273
Maximum Frequency	133.51 MHz
Total Number Pins	500 / 908

Table VI summarizes the lookup performance from the simulation for the IP algorithm selected, MBT or BST, in terms of memory accesses per packet. Note that, for a comparison, this measure is according to a unique packet, while the lookup process in MBT is a pipelining design to obtain higher throughput. The memory requirement and the number of rules that can be stored using the same number of memory blocks is also presented in the Table VI.

TABLE VI. PERFORMANCE EVALUATION FOR IP ALGORITHM

IP Lookup Algorithm	Lookup Memory Accesses (clock cycle)	Memory Space Required	Number of Stored Rules
MBT	1 per packet	543 Kbits	8K rules
BST	16 per packet	49 Kbits	12K rules

It can be seen that MBT realizes a fast lookup process while BST can support a greater number of rules. If the network application requires MBT for fast lookup, an external memory can be used for a rule filter with more than 8K rules. Although BST has notable disadvantages in search speed, a 12K rule filter can be stored using embedded memory, which is attractive for applications with large rule filters.

Our approach holds high flexibility capacity and the architecture can be adapted with different single-field lookup algorithms that obey certain conditions for label method and where the packet header can be segmented.

The work presented in [9] proposed a HyperCuts based algorithm. In this case, the need for syntax alteration and the linear search is critical. However in our system the algorithms are chosen according to the field characteristics.

Although the architecture in [9] supports incremental update, this is performed in hardware and the complexity is high in comparison with the proposed architecture in this paper. According to the results in [9], HyperCuts requires large memory usage. However, it is able to achieve high throughput for 5 fields and is a candidate technique to be implemented in our configurable lookup architecture.

DCFLE groups unique rule field value sets according to the exact matching values. For that, this algorithm uses TCAMs for IP address and ETCAM for the rest of the fields for range comparison. This architecture is implemented in pipelined stages. However, it cannot achieve target throughputs for the current line rate.

For further evaluation, Table VII compares our design against different hardware designs based on 5-field packet classification, assuming a packet size of 40 bytes. In the future, the system must be able to handle a minimum number of 15 header fields to support the current OpenFlow protocol.

TABLE VII. PERFORMANCE COMPARISON

Algorithm	Memory Space (Mb)	Number of Stored Rules	Throughput (Gbps)
Our system with MBT	2.1	8K	42.73
Our system with BST	2.1	12K	2.67
Optimizing HyperCuts [9]	4.90	10K	80.23
DCFLE [4]	1.77	128	16

VI. CONCLUSION

The principal contribution of this research is a highly configurable parallel lookup architecture for IP flow classification. The proposed and prototyped lookup architecture targets primarily SDN systems providing a fast update through software programmability. It offers optimal lookup performance by configuring the best performing set of algorithms for a given type of flow entries. Efficient memory utilization is also achieved by sharing memory resources among multiple lookup algorithms. An example lookup configuration of a multi-bit trie lookup algorithm achieves 133 million lookups per second. Assuming an average IP packet size of 100 bytes, this is equivalent to over 100 Gbit/s link throughput, surpassing lookup targets of emerging high-performance network technology. If configured as binary search trie, a higher rule storage capacity can be achieved. The proposed architecture can be extended with new custom-purpose algorithms in the configuration-set so that the scope of the lookup circuit can be widened to new SDN applications and flow entries. This solution is ideally suited for the evolution of SDN with granular flow classification for emerging SDN applications.

REFERENCES

[1] S. Sezer, S. Scott-Hayward, P. Kaur Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, N. Rao. “Are we ready for SDN?-Implementation Challenges for Software- Defined Networks”. *IEEE Communications Magazine,* Vol 51, pp. 36-43, 2013.

[2] S. Singh, F. Baboescu, G. Varghese, J. Wang “Packet ClassificationUsing Multidimensional Cutting”. *SIGCOMM,* pp. 213-224, 2003.

[3] P. Gupta and N. Mckeown, "Packet classification on Multiple Fields". *SIGCOMM’99,* pp. 147-160, 1999.

[4] S. Sahni, K. S. Kim, “Efficient Construction of Variable-Stride Multi-bit Tries For IP Lookup”. *IEEE SAINT*, pp. 220-227, 2002.

[5] D. E. Taylor and J.S. Turner, “Scalable Packet Classification using Distributed Crossproducting of Field labels”, *IEEE INFOCOM 2005*, Vol. 1, pp. 269-280, 2005.

[6] G. S. Jedhe, A. Ramamoorthy, and K. Varghese, “A scalable high throughput firewall in FPGA”. *IEEE FCCM*, pp. 43-52, 2008.

[7] B. Vmanan, G. Voskuilen, T.N. Vijaykumar “EffiCuts: Optimizing Packet Classification for Memory and Throughput”. *ACM SIGCOMM Computer Communication Review*, pp. 207-218, 2010.

[8] T. Stimpfling, Y. Savaria, A, Belieau, N.belanger, O. Cherkaoui “Optimal Packet Classification Applicable to the OpenFlow Context”. *High performance and programmable networking*, pp. 9-14, 2013.

[9] W. Jiang, V. Prasanna “Scalable Packet Classification on FPGA”. *IEEE Transactions on Very Large Scale Integration (VLSI) System,* pp. 1668-1680, 2012.

[10] Y. R. Qu, S. zhou, V. K. Preasanna “Scalable Many-field Packet Classification on Multi-core Porcessors”. *Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 33-44, 2013

[11] A. Banerjee Thesis: “Improving Data Transport over High-Speed Networks”. University of California Davis. 2007.

[12] H. Song, www.arl.wustl.edu/~hs1/project/filterset. Accessed January 2014.

[13] M. Dixit A. Kale, M. Narote, S. Talwalkar, B.V. Barbadekar, “Fast Packet Classification Algorithms”. *International Journal of Computer Theory and Engeneering,* Vol 4, pp. 1030-1034, 2012.

[14] S. Hsieh, Y. Huang, Y. Yang. “Multiprefix Trie: A New Data Structure for Designing Dynamic Router-Tables”. *IEEE Transaction on Computer*, pp. 693-706, 2011.

[15] P. He, H. Guan, G. Xie, K. Salamatian “Evaluating and Optimizing IP Lookup on Many core Processors*”. ICCCN 2012*, pp. 1-7, 2012

[16] A. Nikitakis and I. Papaefstathiou “A Memory-Efficient FPGA-based Classification Engine”. *IEEE FCCM,* pp. 802-807, 2008.

[17] K. Guerra Perez, X. Yang, S. Scott-Hayward, S. Sezer “Optimized Packet Classification for Software-Defined Networking*”. IEEE ICC’14,* 2014.